PC-Based Oscilloscope

Design Review

Submitted by
Todd M. Buelow
Michael Jendrysik
Chris Justice
Mike McClimans
Brian J. Smith

May99-09

Submitted 12.8.98

Table of Contents

Abstract	3
Background	3
Design Objectives	3
Technical Solutions	4
Hardware Solutions	4
Firmware Solutions	6
Software Solutions	6
Fiscal and Human Resource Budgets	7
Two Semester Schedule	8
Parties Involved	8
Appendix 1: Figures	9
Appendix 2: Two Semester Schedule	12

Abstract

The nature of this project is to research, design, develop, and test a device that will take voltage – time measurements, as an oscilloscope does, and output those results to a PC. This device, called the PC-Based Oscilloscope (henceforth referred to as *the module*), will interface with its host PC via the serial port. The user will interface with the module via a custom-designed Microsoft Windows 95 compatible application. The custom software will present the digitized voltage measurements to the user in the form of a live waveform display.

Background

This project will create a product that will be very useful to the typical college student and the electronic hobbyist. A simple, low-cost oscilloscope is all that is often needed to solve the kinds of problems presented in most undergraduate electronics laboratories and in at-home electronic projects. The PC-Based Oscilloscope will provide that solution.

Design Objectives

The fundamental goal is for the PC-Based Oscilloscope to have basic oscilloscope functionality. The module itself will be self-contained, bearing no external controls. It will have a power cord and a nine-pin serial port connector. In addition, it will bear two BNC connectors for the input channels, with which standard commercial 10x oscilloscope probes may be used. All control will be done with the companion Windows 95 software.

The PC software will have the look and feel of the front panel of a traditional oscilloscope. When communicating with the module via the serial port, the bulk of the traffic will be measured data. However, the host PC software will send messages to the module for changes of trigger conditions data format.

Following are the details of the design.

Hardware Details:

- 2 input channels
- ±25 volt input signal swing.
- 100KHz sample rate
- 32K data memory depth
- RS-232 interface to PC (115,200 baud, 8bit, no parity)
- 12V DC wall power supply
- <1W power consumption
- Host controlled

Software details:

- Voltage vs. time display
- Sampling rate controls
- Voltage scaling controls
- Trigger control
- Individual channel setup
- Serial data and control interface with hardware

Technical Solutions

The design of the product was split into three major areas: hardware, firmware, and software. The most time consuming area is hardware. It was also the area which must be completed ahead of the others, since making changes in hardware methods are much more difficult than making changes in code. Following are detailed descriptions of to-date solutions.

Hardware Solutions

The probe hardware can logically be broken down into several functional blocks. See Figure 1, *Module Hardware Block Diagram*. The following sections will describe the function and purpose of each logical block, as well as the overall theory of operation.

The basic operation of the module is fairly straightforward. Analog signals are converted to digital form, and stored to temporary memory under control of the microcontroller. When the appropriate amount of data is collected, the microcontroller sends the digitized data to the host PC via the serial port. Any configurations changes come to the microcontroller via the serial interface from the host PC.

Instrumentation Amplifier Block

As previously stated, analog input signals are conditioned by the analog instrumentation amplifier block, where common mode noise is removed, and the signals are conditioned for the Analog to Digital Converter Block. The Instrumentation amplifiers chosen also add some level of ESD protection on the analog inputs. The output of this stage feeds directly into the Analog to digital converter stage, covered in the next section.

The physical design of the analog input stage is fairly simple. It consists of a unity gain buffer stage that has a maximum bandwidth of 4 MHz. Signals enter this stage from a 10x oscilloscope probe, giving a divide-by-ten for the input voltage. Because the input voltage swing is ±5 Volts, and because of the +2.5V offset voltage, the input voltage range is ±25 Volts at the probe tip. In the interest of keeping a simple design, there is no hardware gain adjustment. Hardware gain adjustment was considered for a time, however it was left out of our final design at the advice of our team advisor, Dr. Carlson.

Analog to Digital Converter Block

The Analog to Digital Converter Block is simply a two channel, 8-bit analog to digital converter. The converter that we selected was chosen for it's very high bandwidth among other features. The ADC is controlled by the microcontroller, and can either sample both channels at the same time, or either channel alone. The ADC reads an analog voltage, and converts the voltage to an 8-bit digital quantity. The data gets latched onto the 8-bit wide data bus to be stored in memory and is also an input to the Digital Trigger circuit.

Digital Trigger

The Digital Trigger circuit reads 8-bit trigger values off the data bus, and latches them onto one side of its 8-bit digital magnitude comparator. Once a trigger value has been latched onto the digital trigger latch, it is compared with data on the bus coming from the ADC. When any of three trigger conditions are met, the appropriate pin is asserted and the microcontroller is notified of a trigger status. The three trigger conditions are: rising edge past a voltage threshold, falling edge past a threshold, and an equivalence condition. The equivalence condition is provided for slow signals, and will probably not be used in most cases.

The digital trigger was not originally part of the probe design. The digital trigger was added to lighten the processing load on the microcontroller and to achieve a higher sampling rate, should such conditions be requested by the host. The digital trigger consists of three CMOS logic gates, an 8-bit latch, and two parts that make up an 8-bit digital magnitude comparison circuit. This circuit block is not necessary to the operation of the probe, as the microcontroller could implement this functionality in software. As mentioned earlier, this feature was added to allow faster trigger recognition, and to lighten the processing load of the microcontroller. Refer to the microcontroller section for more information of the bottle-neck created by the slower microcontroller.

Memory Block
The Memory Block allows the 8-bit digital data to be buffered quickly during acquisitions, and then output slowly to the serial port. For ease of design, only 32k of memory is provided. This is assumed to be ample memory to store meaningful data for two channels. The memory is also on the 8-bit data bus, and functions in two modes. The first memory mode is where it is not used at all. The ADC data can be sent directly out the serial port to the host PC, never being stored in memory. This allows a continuous stream of data to be output to the host. This is a slower method of data capture, and is limited by the 115,200 baud serial port transmission speed limit. The other mode is where data from the ADC is buffered in the memory until it is full, and then dumped to the serial port. This mode will obviously result in faster sampling rates.

Asynchronous static RAM was used because it is very fast, doesn't require the overhead of memory refresh, and allows simple writing interface. The 32k memory space requires 15 address lines which are controlled by the Address Counter Block. Address space is linear, and probe data is stored sequentially. This means that when two channels are being sampled, the data is stored in an alternating fashion (channel one, channel two, channel one, channel two...). The microcontroller has the option of sending this data to the host in this raw format, or of sending one channel's data stream at a time. Chip enables and mode selection control is handled by the microcontroller.

Address Counter
The Address Counter was added to take some of the processing load off of the microcontroller, by outputting the correct 15-bit address to the Memory Block. Implementation of the address counter was achieved through the use of discrete CMOS building blocks. This method will be shown to function somewhat like a port expander, keeping the number of output pins from the microcontroller smaller and the firmware simpler.

The Address Counter can either be reset, count up, or count down (or do nothing). Its output, the memory address, is incremented by a clock signal from the microcontroller. The address counters' exact implementation is still being worked on, but the general operation will follow these descriptions. One implementation option that is currently being explored is allowing the address counter to function with less intervention from the microcontroller. This would ideally be accomplished by using the inputs of the trigger circuit and static control pins from the microcontroller to operate the Address Counter. This would eliminate the need for the microcontroller to toggle the clock input to the Address Counter, an operation that takes two instruction cycles.

Microcontroller Block
The microcontroller is the heart of the system, controlling the states and operation of almost all other blocks of the system. The microcontroller selected is the Microchip PIC16C74A. This microcontroller was selected primarily for it's ease of code development, good documentation, low power dissipation, simple instruction set, and features that make using a serial port interface easy.

The majority of the microcontroller code is written in C, with some of the speed-critical sections being manipulated in assembly language. For more information on the operation of the microcontroller firmware, see the firmware section of the *Technical Solutions*.

The microcontroller is connected to the 8-bit data bus. The microcontroller latches out trigger values and also reads data from the ADC or memory on this bus. The microcontroller determines which device has control of the bus, and tri-states all of the other devices. In the event of a trigger condition, the microcontroller directs all of the ADC data to the memory, and after it has all been stored, slowly sends it out the serial port.

The microcontroller has an integrated UART (Universal Asynchronous Receiver Transmitter) that interfaces with the host PC serially. This UART is fairly simple in its implementation, but still flexible. The serial port will be operated in fixed-baud rate mode in order to simplify design. The microcontroller transmits data and status information to the host, and receives commands from the host.

The microcontroller is clocked from an external 20mhz crystal oscillator. At this clock frequency, the PIC executes 5 MIPS. The PIC architecture is pipeline RISC, so all instructions take one cycle, except for *jump*'s which take two.

Serial Translator Block
This block is simply a standard RS-232 format serial port, which receiver data from the microcontroller's UART. The serial translator block voltage shifts the signals from the microcontroller, as well as adding some ESD protection. The serial data format is 8 data bits, 1 stop bit, and no parity bit. One of the suggestions from an early design review was to eliminate this part in favor of simple 5-volt logic levels from the microcontroller. This was ruled out in favor of the increased ESD protection and increase in compatibility which the RS-232 device could bring with it.

Power Supply
The power supply takes 12 Volt DC input and steps it down to the system voltage of 5 volts. This step down is accomplished by a pair of switching regulators. The first regulator reduces the 'noisy' 12 V power input to 5 V, and filters out much of the incoming noise. One challenge is that the switching regulator operates between 200 and 300kHz, which is a potential liability for analog noise. The switching regulator approach was chosen over linear parts because the favorable level of heat generation and better efficiency. The power supply design is designed to be able to output 300mA on the 5 volt supply.

The negative supply is used only for the analog input amplifiers in order to facilitate the reading of negative voltages. The negative supply takes the regulated 5 volt supply and inverts it. This device is also a high efficiency switching supply and was chosen for its low power dissipation and low heat generation.

Test Signal Generator
This device is simply a square wave generator that allows the user to test the system against a known signal for calibration of probes. The signal generator is designed to output a 1kHz square wave between 0 to 5 volts.

Firmware Solutions

The firmware portion of the project consists of mixed C and assembly code that will reside on the module's microcontroller's onboard re-programmable memory. Although the majority of the code is still developmental, the algorithms for the two major data collection methods are complete. These algorithms are detailed in Figure 2, Firmware Algorithm Flow Diagram.

Although this diagram is somewhat self-explanatory, an important note is the continuous check for commands from the host PC (detailed below). This will ensure that the user will be able to switch modes on the fly and only miss a portion of a full memory load in the worst case. This is somewhat negligible, however, due to the fact that a mouse click could not be executed with the accuracy to stop measurements at any certain time.

Software Solutions

The module's companion software is being developed using Microsoft Visual C++. Upon completion, it will be compatible with Windows 95. A screen shot of the application's user interface follows in Figure 3, *Application User Interface*. Once the software and firmware are completed they will have a defined data structure to ensure clear communication between the two modules. This structure is still being finalized.

Communication work will all be done at the user level (with the rest of the code), using MS Visual C++'s ReadFile and WriteFile commands. These commands are performed on handle created with the CreateFile function, sending commands down the long-established driver hierarchy for the serial port. These functions may be used for asynchronous data communication, but since the module cannot handle multiple threads, this would be unnecessary.

Fiscal and Human Resources Budgets

Function	Projected Effort (person-hours)	Actual Effort (person-hours)	Projected Cost	Actual Cost
Hardware Design				
Function Definitions	20	25	$ -	$ -
Initial Research	30	35	$ -	$ -
Design / order parts	60	35	$ 30.00	$ -
Board Layout	50	40	$ 50.00	$ -
Integration	35	0	$ -	$ -
Testing	65	0	$ -	$ -
Subtotal	260	135	$ 80.00	$ -
Software Design				
Function Definitions	25	20	$ -	$ -
Initial Research	35	30	$ -	$ -
Coding	240	100	$ -	$ -
Debugging	85	0	$ -	$ -
Subtotal	385	150	$ -	$ -
Firmware Design				
Function Definitions	25	20	$ -	$ -
Research	35	25	$ -	$ -
Development	75	20	$ -	$ -
Subtotal	135	65	$ -	$ -
Documentation				
User Manual	20	0	$ -	$ -
Communication Specs	15	0	$ -	$ -
Process/ History	20	0	$ -	$ -
Subtotal	55	0	$ -	$ -
Design Reviews				
Review #1	10	15	$ -	$ -
Review #2	10	12	$ -	$ -
Subtotal	20	27	$ -	$ -
Deliverables				
Project Plan	10	10	$ -	$ -
Project Poster	20	25	$ 15.00	$ 50.00
Design Review	20	0	$ -	$ -
Subtotal	50	35	$ 15.00	$ 50.00
Total	905	412	$ 95.00	$ 50.00
Available resources	900	450	$ 100.00	$ 50.00

Two-Semester Schedule

Please see Appendix 2: Two-Semester Schedule

The breakdown of team members' responsibilities for the following semester is as follows.

Mr. Buelow: Firmware development, software communication development.
Mr. Jendrysik: Software communication and data-intensive/graphical software development.
Mr. Justice: Hardware development and testing; strategic planning.
Mr. McClimans: Hardware development and testing; resource allocation.
Mr. Smith: Firmware development, hardware development and testing.

The following areas are shared responsibilities among the team members: technical writing and proofing, algorithm development.

Parties Involved

Team Members:

Todd M. Buelow
2520 Chamberlain
Ames, IA 50014
268-9648
tmbuelow@iastate.edu
EE 461

Mike McClimans
2209 Barr Dr.
Ames, IA 50010
233-9139
mmcclima@iastate.edu
EE 461

Michael Jendrysik
Lyon 218 Barker
Ames, IA 50013
296-4501
dourden@iastate.edu
CprE 481

Brian J. Smith
Helser 4814 Richey
Ames, IA 50012
572-2870
brjsmith@iastate.edu
EE 461

Chris Justice
4315 Maricopa Dr. #1
Ames, IA 50014
296-1694
cjustice@iastate.edu
EE 461

Team Advisor:

Dr. David Carlson
Associate Professor of Electrical and Computer Engineering
372 Durham Hall
Iowa State University
294-7695
dcarlson@iastate.edu

Appendix 1: Figures

Following pages.

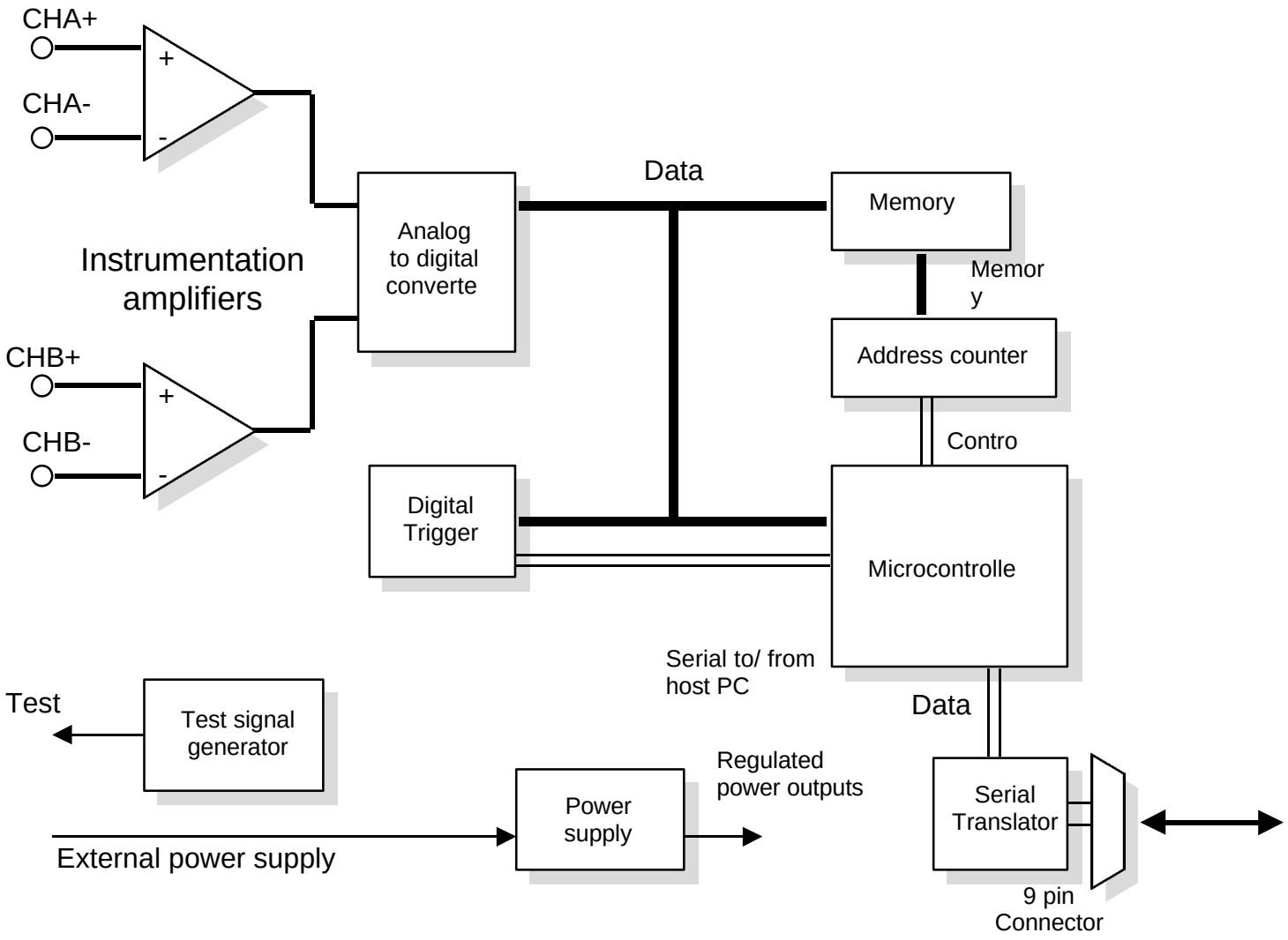

Figure 1. *Module Hardware Block Diagram*

CDJ 10/14/98

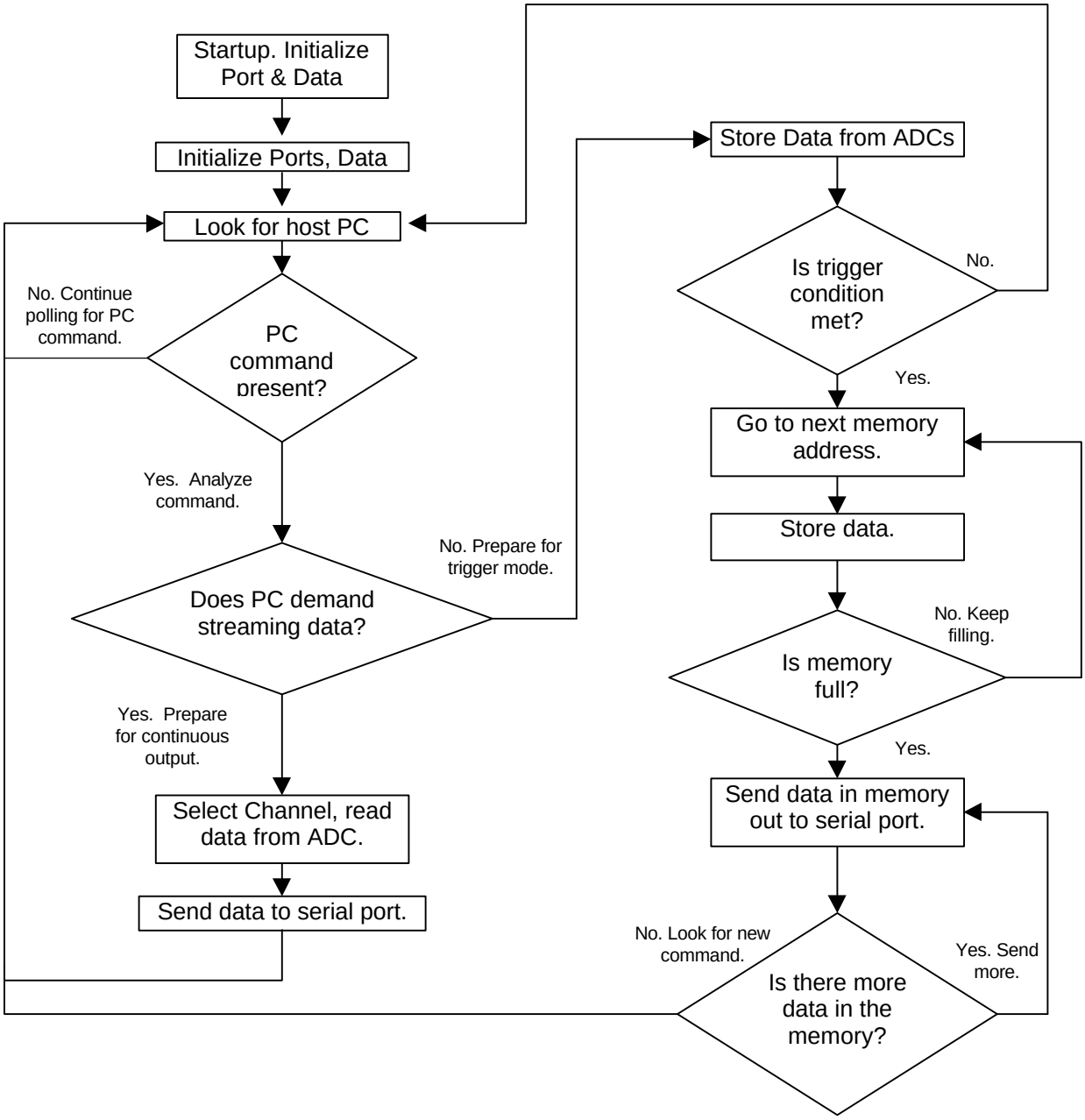
Startup. Initialize Port & Data
Initialize Ports, Data
Look for host PC
No. Continue polling for PC command.
PC command present?
Yes. Analyze command.
Does PC demand streaming data?
No. Prepare for trigger mode.
Yes. Prepare for continuous output.
Select Channel, read data from ADC.
Send data to serial port.
Store Data from ADCs
Is trigger condition met?
No.
Yes.
Go to next memory address.
Store data.
Is memory full?
No. Keep filling.
Yes.
Send data in memory out to serial port.
Is there more data in the memory?
No. Look for new command.
Yes. Send more.

Figure 2. *Firmware Algorithm Flow Diagram*

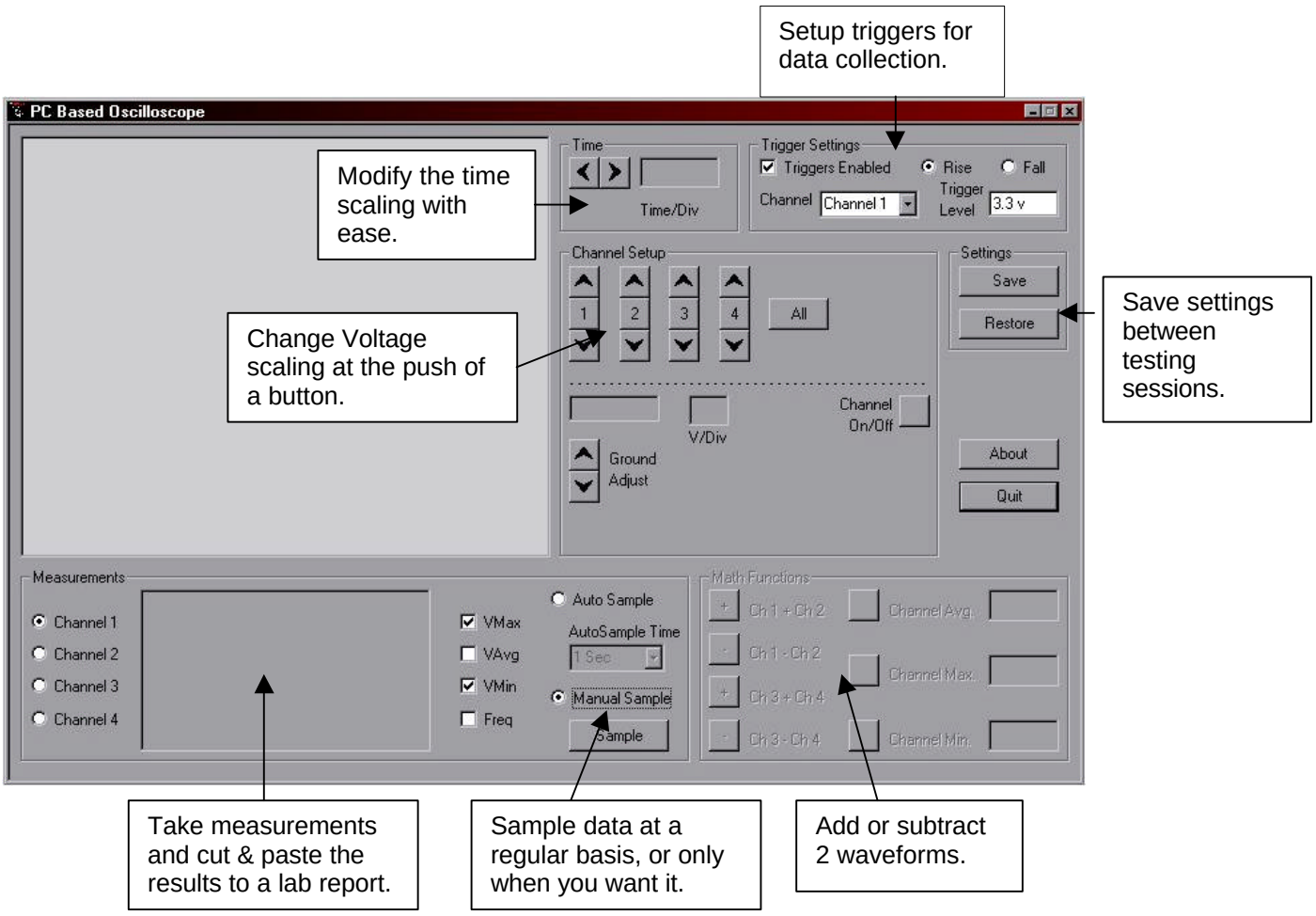

Figure 3. *Application User Interface*